AdaptAV: Continuous Adaption of Vision Models for Autonomous Vehicles Using Cloud-based Oracle

Yuheng Zhu, Dhruva Ungrupulithaya, Boluo Ge, Man-Ki Yoon
Department of Computer Science
North Carolina State University

***Abstract*—Deploying vision perception models in autonomous vehicles requires that we prioritize inference speeds, resulting in a model with shallower architectures and lesser model parameters (i.e., more pruned). Such small models do not generalize well, which could result in poor performance when encountered with novel scenarios. We propose a system that overcomes this by continuously retraining the vision models on the cloud with data uploaded by vehicles. We leverage the abundant compute resources, including machine learning accelerators, of the cloud to run a highly-accurate oracle model that will guide the retraining process of the on-vehicle model. This newly trained model is transmitted to the vehicle over the network and is utilized by the vehicle for perceptions, leading to improved inference accuracy over time.**

I. Introduction

The proliferation of autonomous vehicles (AVs) has been one of the most significant trends in vehicular technology with automakers investing heavily [1]. Perception models are key to AVs' success and work by taking inputs from an array of sensors like cameras and LiDARs to assess the surrounding environment and make relevant decisions. The development of these intelligent algorithms is highly data-driven and is a result of the collection of millions of hours of driving data in a variety of unique environmental conditions and driving scenarios and training the models on these extensive datasets.

Although the models are tested vigorously to ensure that they perform well in a variety of challenging conditions, optimal performance is not always guaranteed. Poor performance can be attributed to two primary reasons: a suboptimal model and/or inadequate comprehensive training. Due to the constrained computational and memory resources available in vehicles, the perception models running in these systems may not be as precise as the most advanced models currently accessible. These models are instead optimized for resource efficiency, resulting in the model performing inference quickly to meet the real-time requirements of an autonomous vehicle. Thus, the perception model performs poorly when it encounters novel and challenging scenarios not included in the training dataset, leading to inferior decisions. For instance, Fig. 1 shows the detection of a larger model (Faster R-CNN [2]) and that of a smaller model (SSD-MobileNet [3] [4]) that would run on the vehicle, both of which are trained on the Waymo Open Dataset [5]. There is a significant difference in the inference outputs of the two models, with the smaller MobileNet model missing some key objects such as the pedestrian crossing the road. This poor perception performance, if not addressed, may lead to catastrophic consequences [6] [7].

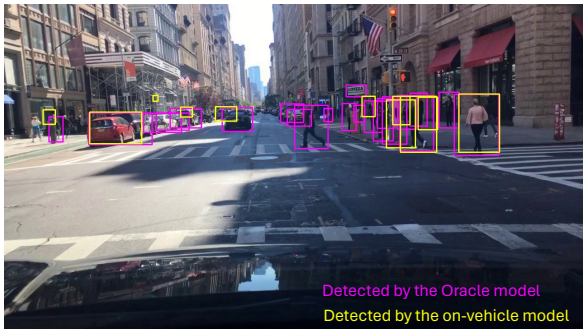

Fig. 1. A scenario where the vision model deployed in the vehicle fails to detect multiple important objects. The purple bounding boxes correspond to the detection of the larger Faster R-CNN model, and the yellow bounding boxes correspond to the smaller SSD-MobileNet model.

The conventional approach would be to deploy test vehicles to collect more data and use it to train and develop a better perception model. This improved model would then be updated when the vehicle comes to the service center as a software update, or if supported, through an over-the-air (OTA) update. However, the process of data collection and development of the updated model would take significant amounts of time and resources and still may not address all the shortcomings of the previous model [8].

Drawing from the above insights, we propose AdaptAV, a closed-loop software architecture that continuously improves and updates the model running in the vehicle. We achieve this by leveraging the power of computing on the *cloud*, which enables us to apply a complex, powerful perception model to flag erroneous or poor decisions of the on-vehicle model. These *oracle* models are developed to prioritize inference accuracy rather than speed, which allows us to use them to steer the (re)training of the small model that runs on the vehicle. The vehicle continuously logs the vision data and pushes it to the cloud in a bandwidth-efficient manner by using popular video compression techniques, allowing us to deploy this system even with limited network connectivity.

In this paper, we make the following contributions:

- We present AdaptAV, a closed-loop software framework that continuously updates the vision model running in the vehicle using the cloud-side oracle model and the most recently captured image stream;
- We propose frame sampling techniques that can reduce overfitting of the vision model;
- We analyze the AdaptAV design parameters and their impacts on ML inference quality; and

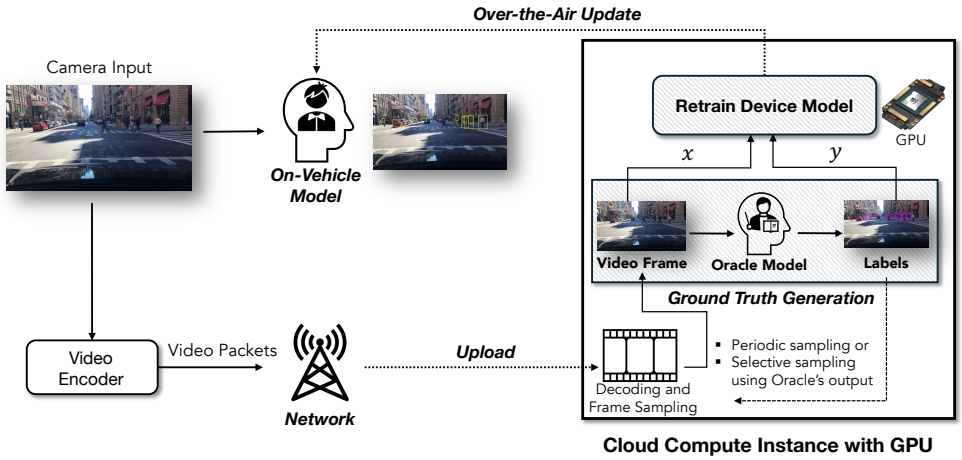

Fig. 2. AdaptAV continuously updates the vision model running on the vehicle. The image stream used by the vision model is compressed and uploaded to the cloud. A cloud instance with powerful compute with machine learning accelerators uses the data to retrain the model. The retrained model is then deployed back to the vehicle, closing the loop.

- We show the practicality of AdaptAV by applying it to a prototype vehicle integrated with a public cloud.

II. Related Work

For visual object detection, existing methods can be divided into one-stage approach, represented by YOLO [9] [10] [11], SSD [4], and two-stage approaches represented by R-CNN and its various variants [12] [2] [13]. One-stage approaches can generate object location and confidence in only a single stage, which ensures the inference speed and thus is more suitable to be deployed on vehicles. Two-stage models need to generate region proposals first and then generate final detection based on the region proposals. It offers better accuracy than one-stage models but with higher latency.

One common approach to using cloud computing resources to alleviate computational power constraints of device is to apply an early exit strategy on the device model, which reduces the device-side inference latency by implementing some scheduling procedures to split some layers of the inference model to the cloud or edge computing platform [14] [15]. For example, Edge AI [16] implements a DNN-based collaborative inference framework for edge computing, which uses early exit to distribute computing loads across devices and edge servers, but such approaches rely on consistent model architectures, making the framework overly coupled with the inference model architecture.

The other approach is that the device model completely undertakes all visual inference tasks for the vehicle and uses the cloud model as a teacher to transfer its generalization capability to the device model. This kind of *knowledge distillation* approach ensures the decoupling of the model architectures between the device and the cloud. However, it requires the visual data collected on the vehicles to be sent to the cloud for inference, which leads to higher demand on network bandwidth. Edge Compression [17] proposes the use of compressive imaging (CI) cameras in addition to specialized energy-efficient deep neural networks in the cloud and edge servers. However, this approach would require the use of less prevalent and more expensive CI cameras, as well as a complete redesign of the other vision-based software modules in the vehicle.

Ekya [18] is a similar framework for continuous retraining of a vision model. But, it uses edge servers instead of the cloud because the bandwidth required to upload raw camera data is too high. More importantly, their edge servers also perform vision inference, unlike our approach, which performs inference locally on AV devices. Khani et al. [19] propose an on-the-device frame sampling technique to reduce the network bandwidth requirements. For the sampling, it utilizes the device model's results, which can be problematic since these results may not be accurate enough to help determine the samples relevant for the retraining process. Our framework is novel in that it leverages an oracle model running on a cloud platform to continuously improve vision models.

III. AdaptAV: Continuous Adaption Using Cloud-based Oracle Model

A. High-level Idea of AdaptAV

Due to limited computing power, vehicles cannot deploy extremely large models with complex architectures. Although using these models for real-time inference in the vehicle is impractical, their high accuracy makes them suitable as *oracle*. The oracle model, trained on an extensive dataset, generalizes better than the smaller, shallower models running on the vehicles. Consequently, an oracle model performs significantly better when encountering new scenes not seen during training. As shown in Table I, the Faster-RCNN model (the oracle model) achieves a significantly higher mean average precision (mAP)[1] of 52% compared to the smaller SSD-Mobilenet model that runs in the vehicle (mAP of 23%). Both models were trained on 35,712 images from the Waymo Open Dataset [5] and tested on a separate subset of 3,968 images.

[1]It is an evaluation metric for object detection models that summarizes the precision-recall curve as the area under the curve across all confidence thresholds. The *precision* of a model is calculated as TP/(TP + FP) where TP is the number of true positives, and FP is the number of false positives. The *recall* of a model is calculated as TP/(FP + FN) where FN is the number of false negatives. A higher AP indicates a higher accuracy and vice versa.

TABLE I
MEAN AVERAGE PRECISION FOR ORACLE MODEL AND DEVICE MODEL

Class	Faster R-CNN	SSD-Mobilenet
person	0.45	0.17
car	0.72	0.49
bicycle	0.41	0.18
sign	0.48	0.09
Average	0.52	0.23

Drawing from these results, we hypothesize that the predictions from the oracle model can be treated as the *ground truth* for retraining the on-vehicle model. AdaptAV is a closed-loop pipeline that streamlines this process as shown in Fig. 2. Autonomous vehicles typically have limited computing and memory resources dedicated to performing time and safety-critical tasks essential to their functioning. Leveraging cloud storage and computing provides us with near-infinite memory and powerful computing capabilities to implement AdaptAV. It continuously streams the camera images to the cloud, where the cloud-side compute instance runs the oracle model to process the images. The results of the oracle model, i.e., the ground truth, are used to retrain the device model, which is then sent back to the vehicle. AdaptAV hence continuously improves the on-vehicle model, making it more robust to new scenarios.

B. Video Compression

As shown in Fig. 2, the raw image stream is fed periodically to AdaptAV as it is captured by the camera. Although cloud resources offer significant advantages, the network bandwidth can easily become the bottleneck because transmitting the raw image stream to the cloud may be prohibitively slow. Camera sensors used in autonomous vehicles (such as MIPI CSI cameras) generate high-resolution images at a high rate to capture the finest details of the environment. Unlike webcams, they do not have built-in codecs, so the raw frames generate an enormous amount of data. Suppose the vehicle has a single 3-channel Full HD camera (1920 x 1080 x 3) recording at 30 Hz. Then, we would need a network capable of upload speeds of approximately 1.4 Gbps.

AdaptAV tackles the challenges by compressing the raw frames into a video stream, which consumes much less network resources to transmit. Specifically, the raw frames are encoded into a sequence of video `packets`, which are queued and pushed to the cloud by a separate thread. For the compression, we use `FFmpeg` [20] as a backend, which allows us to use a variety of video codecs. In this paper, we utilize the ubiquitous H.264 codec (`libx264`) for video compression, which is the most common encoding format in various domains such as online streaming and digital television. Its high compression efficiency is demonstrated by the Waymo dataset in our evaluation. This dataset, consisting of 200 subsets and requiring approximately 286 GB of storage, can be compressed into a video of mere 4.2 GB, saving storage over 98%.

C. Frame Sampling

The compressed image stream transmitted to the cloud is extensive. If the camera is emitting data at a rate of 30 frames per second, 108,000 images are generated in just 1 hour of driving. Using all these images to retrain the device model would take a large amount of computing resources. Moreover, we risk *overfitting* the model by using all the images, i.e., the model will *memorize* rather than *generalize*. Fig. 3 shows that when using all of the BDD100K dataset [21] (which will be introduced in the next section) for training, overfitting starts to occur. To overcome the problem, we propose the following *frame sampling* techniques.

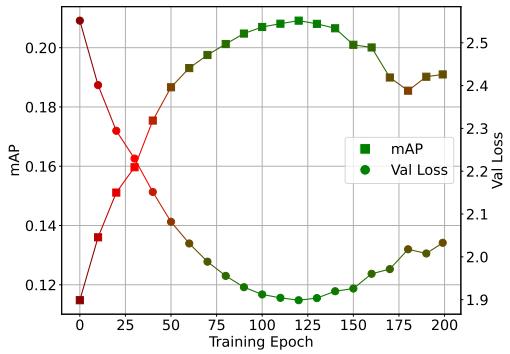

Fig. 3. A model is likely to overfit when using frames captured/sampled at a high rate. Hence, a frame sampling is necessary.

a) Periodic sampling: This is the baseline approach in which we periodically sample an image in a sampling period across a chosen window of the recorded data. For instance, we can choose the first image for every n images, effectively reducing the training dataset to $1/n$ of its original size compared to using all incoming images.

b) Density-based sampling: Another approach is to take into account the oracle model's prediction from each of the images. As an example, particularly for object detection tasks, we propose a method that selects the frame with the *highest* number of objects in each sampling period (with ties broken by randomly choosing one), as illustrated in Fig. 4. The number of objects in each image is obtained from the prediction by the oracle model. The idea behind this approach is that for a sequence of images captured within a short period of time, their objects and predictions should be similar. If we want to downsample this set of images, the most rational way to do so is by selecting the frame with the *most* information, which can be implied by the number of objects present.

c) Event-based sampling: For event-based sampling, the criterion for selecting a frame is the occurrence of an *event*, which can be defined by changes in the types or behavior of detected objects. For example, as shown in Fig. 5, the occurrence of a new object not previously present or the disappearance of an existing object is considered an event. Similar to the density-based sampling, this approach also utilizes the oracle model's predictions, incorporating not only object detection results but also *tracking* information. Frames are selected when preset events are detected. The rationale behind the idea is that when the same objects appear in a series of frames without any significant change, they do not deliver much 'new' information. This sampling approach is designed to ensure that only frames with *meaningful changes* are selected, maintaining the relevance

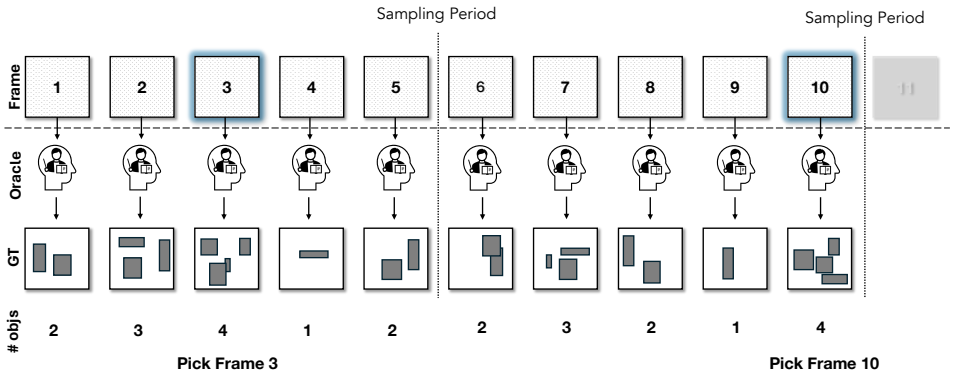

Fig. 4. Density-based frame sampling. During each sampling period, the one that has the largest number of detected objects is selected.

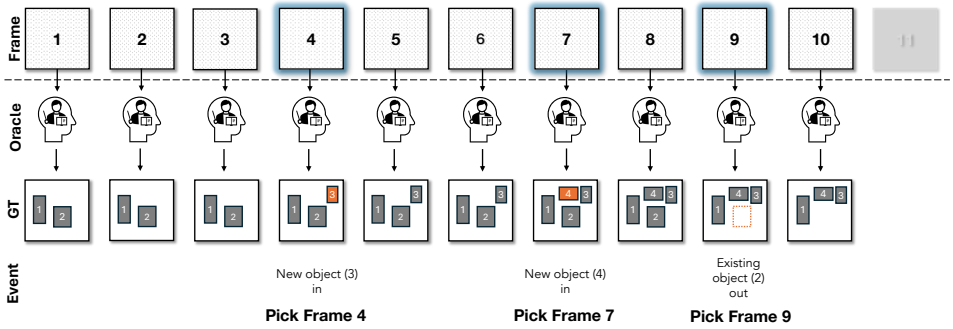

Fig. 5. Event-based frame sampling. An event is defined by the appearance of a new object or the disappearance of an existing object.

of the sampled frames. In our implementation, we use SORT [22], a real-time tracking algorithm that tracks each unique object in continuous frames and generates a unique identifier for each of them. By using it to track the motion of objects in frames, we detect when each object first enters and leaves the frame and mark it as an event.

If available, one could instead detect events based on other sensor inputs, such as vehicle turns, sharp braking, or the presence of other objects at relatively high speeds. The proposed sampling approaches are critically different from existing techniques such as [19] in that we leverage the *oracle* model's results, not the device model. Using the device model's inference outputs risks relevant objects going undetected or being misclassified during the retraining process.

Note that *filtering* differs from sampling. For instance, Reducto [23] is an image filtering mechanism running directly on a camera system-on-chip that filters out irrelevant frames for the perception models at the camera level. Mistakes made by the filtering algorithm may drop camera frames for a long period of time, which can be a critical hazard to safety-critical systems like autonomous vehicles.

IV. Evaluation

A. Setup

1) Platforms: Our implementation is based on an NVIDIA AGX Orin Development Kit [24] that we run on a small-footprint vehicle (approx. 3 ft x 6 ft) shown in Fig. 10. It features a 12-core ARM Cortex-A78AE CPU, each core running at 2.2 GHz, a 2048-core NVIDIA Ampere GPU, and 64 GB of memory. The software system is configured with NVIDIA JetPack SDK v5.1.2, which includes Linux for Tegra 35.4.1, CUDA v11.4, cuDNN v8.6.0, and TensorRT v5.1.2. For the prototype unmanned vehicle (detailed in Section IV-C), we utilize Google Cloud [25] Storage Bucket where video packets are uploaded to. We use a Google Cloud virtual machine instance `a2-highgpu-1g` for running the Faster R-CNN oracle model and retraining the on-device model. It has 12 virtual CPUs, 85 GB of memory, an NVIDIA A100 (40 GB) GPU, and operates Debian 11.1 and CUDA v12.2.

2) Datasets:

- Waymo Open Dataset [5]: We choose the first 200 driving video records out of the 2D object detection validation set from Waymo Open Dataset v1.4.2. The subset consists of 39,681 frames with a resolution of 1920 x 1280 captured at 10 Hz. It is used only for the *initial* training of our object detection models.
- BDD100K [21]: MOT-2020 Images set from BDD100K serves as a simulation of actual video stream. The frames are captured at 5 Hz with 1280 x 720 resolution. The dataset comprises 200 driving video records (approx. 200 frames per record), amounting to a total of 39,973 frames.

3) Models:

- Faster R-CNN [2]: It is an object detection model that uses a region proposal network to classify object proposals and predict bounding boxes within a single pass through the network. We utilize it as the *oracle* model for generating the most accurate detection results possible. The model contains 41,092,136 parameters, and the model file occupies 161,916 KB of disk space.

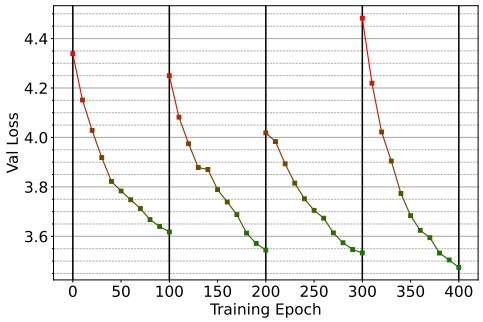

Fig. 6. The validation loss during the continuous retraining of the SSD-Mobilenet model. There are a total of 4 retraining windows.

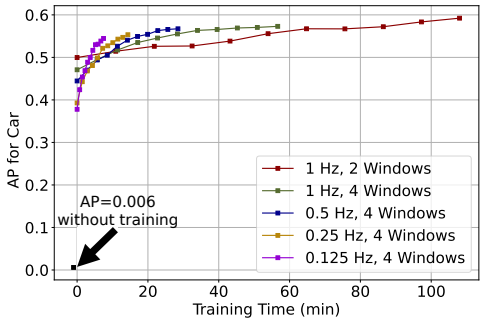

Fig. 7. Model accuracy (AP for the car class) v.s. retraining time for different hyperparameter configurations. The number of windows refers to the number of retraining windows into which the BDD100K training set is divided equally. Each line is cut at the point where the y-axis value converges.

- SSD-MobileNet [3] [4]: It is a fast and efficient object detection model that combines Single Shot Multibox Detector with a MobileNet architecture, making it ideal for real-time applications on resource-constrained devices. In our work, it is deployed on the vehicle to perform real-time detection on camera inputs as the *device* model as mentioned earlier. The model contains 3,206,976 parameters, and the model file size is 27,641 KB.

On the Orin device, while the SSD-MobileNet takes approximately 5 ms to make a single inference, the Faster R-CNN model takes 130 ms. This shows that the oracle model is too large to be run on the on-vehicle device.

B. Results

We evaluate AdaptAV on the BDD100K object detection dataset mentioned above. The dataset is split into the training set (90%) and the validation set (10%). Recall that both the *oracle* model on the cloud and the *device* model deployed on the vehicle are pre-trained with the Waymo dataset and thus have never seen the BDD100K dataset.

For the retraining process, we consider the following two hyperparameters that affect the retraining speed and quality:

- **Retraining sample rate**: Cameras with a high sampling rate will produce too many images, which can result not only in an elongated retraining process but also overfitting, as discussed earlier. Hence, it is necessary to downsample the acquired images to reduce the size of the retraining set.
- **Retraining window size**: It represents the interval between successive retrainings. Similar to the retraining sample rate, a retraining window size that is too large will lead to unnecessarily long retraining times, while a size that is too small may result in the model failing to accurately learn features within the corresponding retraining window.

As mentioned above, we use the BDD100K dataset for the continuous retraining of the SSD-Mobilenet model. We first equally divide the BDD100K dataset into four retraining windows. Hence, each window consists of about 9000 images; one can imagine the on-vehicle camera capturing 9000 images. Then, AdaptAV applies a frame sampling technique, which defaults to periodic sampling, and then trains the model to convergence on each window using the preset retraining hyperparameters. Fig. 6 shows the validation losses over the whole

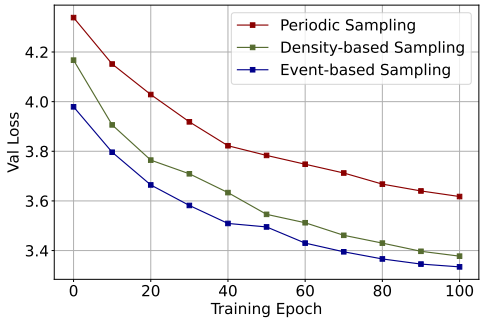

Fig. 8. The impacts of different sampling approaches on the validation loss.

retraining steps. Note that at the end of each retraining window, the updated model is deployed (updated to the device). Hence, what will be visible from the on-vehicle model are the final loss values at the end of each window (i.e., 100th, 200th, 300th, 400th). As we can see, the loss of the model decreases as the retraining process repeats, which indicates that its detection accuracy improves with the continuous retraining. This is further emphasized in Fig. 7, where the accuracy of the model (y-axis) when directly tested on the BDD100K dataset (represented by a single black point) is a mere 0.6%. The model, having never encountered the scenarios in the BDD100K dataset, performs extremely poorly. Retraining the model by just a single epoch, represented by the data points at a time near 0 min (x-axis), leads to a drastic improvement in the model performance, with continuous retraining ultimately yielding an AP of nearly 60%.

Fig. 7 also evaluates the effect of different retraining window sizes on the model retraining. As can be seen, although the model can converge quickly even with a very low sampling rate (small training set) the accuracy decreases; on the contrary, a larger training set obtained with a higher sampling rate results in a higher accuracy of the model but the time required for convergence is also longer. In practice, one needs to select proper hyperparameters according to the computing resources on the cloud and the sampling rate of the vehicle camera, so that the frames for retraining are not sampled too fast or slow.

Fig. 8 shows the impacts of periodic sampling, density-based sampling, and event-based sampling on the validation loss during retraining. Compared to the periodic sampling, the two selective samplings help improve the model accuracy.

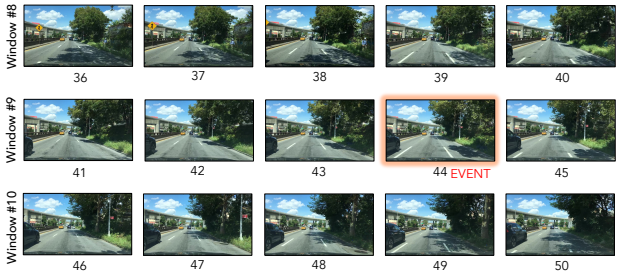

Fig. 9. The blue-colored vehicle on the left of the scene is first detected by the Oracle model at Frame 44. Thus, it is the only frame sampled by the event-based method. The density-based method samples three frames from these three windows, each with a length of five, although the scenes are similar.

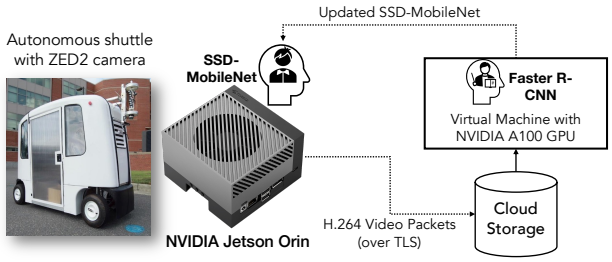

Fig. 10. AdaptAV implemented on a small autonomous shuttle interacting with a public cloud platform. The vehicle uses the NVIDIA Jetson Orin as its computing platform and uses a ZED2 camera inputs for its vision tasks.

As discussed in the previous section, these selective sampling techniques improve the quality of the frame samples by selecting the ones that contain more/better information than others. Meanwhile, the event-based sampling performs the best out of the three approaches overall. We can attribute this result to the fact that both periodic and density-based methods are forced to sample *at least* one frame during each sampling period. This can result in sampling *redundant* frames, such as when the scene is static or has low dynamics (e.g., due to the ego vehicle moving slowly or being stopped), which leads to consecutive periods sampling very similar frames. Fig. 9 illustrates such a situation. These redundant frames contribute little new information during retraining. Conversely, in high-dynamic scenes, both periodic and density-based methods are limited to sample *at most* one frame during each sampling period. As a result, they may miss frames that would have been informative if sampled. The event-based sampling is not constrained by the sampling period; it can sample more or fewer frames depending on the scene dynamics, which makes it suitable for mobile systems like vehicles.

It is important to note that both the density and event based sampling techniques overcome the overfitting issue that is prevalent with continuous training of models. However, implementing the event-based sampling requires additional computing resources to run an object-tracking algorithm. Hence, one may want to opt for the density-based method if computing power is limited. Although it is not as powerful as the event-based method, it performs better than the naive periodic sampling technique while also preventing overfitting.

C. Usecase

We applied AdaptAV to a small-scale autonomous shuttle to continuously improve the SSD-MobileNet object detection model running in the vehicle, as shown in Figure 10. The NVIDIA Jetson Orin device running on the shuttle uses a ZED2 stereo camera [26] for image inputs. The system ended up taking approximately 14,000 frames at a rate of 15 frames per second, which were compressed and sent individually to the cloud where we applied the periodic sampling technique with a window size of five. The vehicle's object detection model was re-trained with the ground truth generated by the Faster-RCNN model in the cloud over four retraining windows. Figure 11 shows the validation loss curve for each of those retraining windows. From the final validation loss in each window, we can observe that the on-vehicle model's accuracy improves as the retraining happens. Both the SSD-MobileNet and Faster R-CNN model were initially trained on the Waymo Open Dataset.

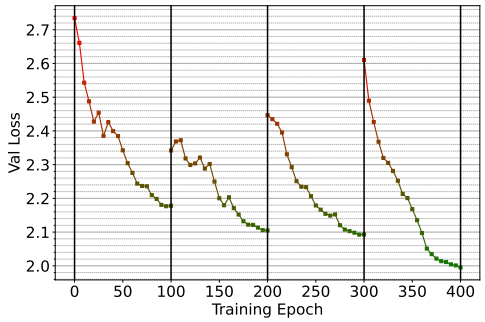

Fig. 11. Validation loss curve for the object detection model running in the autonomous shuttle.

The cloud platform greatly accelerates the inference speed of the oracle model. For Faster R-CNN, the average inference time on the NVIDIA Jetson Orin device for one single frame captured by the shuttle reaches about 130 ms, while it takes only 28 ms for the cloud computing platform. The model training is also significantly accelerated by the cloud as well; for 10 epochs of training for the same retraining window size, it takes 422 seconds to finish on our in-lab workstation (with Intel i9-13900F CPU, 64 GB memory, and NVIDIA GeForce RTX 4090 GPU), but only 178 seconds on the cloud.

D. Discussion

- Vehicle data security and privacy: Vehicles' vision data carries inherent privacy risks. Unauthorized access to this information could occur through network interception or compromise of cloud storage systems, potentially leading to severe breaches of individual privacy. Our usecase presented in Section IV-C ensures the privacy and integrity of the data in transit and at rest by leveraging Transport Layer Security (TLS) and storage encryption, respectively. Most cloud platforms provide these security measures. Major cloud providers also offer *confidential VMs* [27], which enables the confidentiality and integrity of data *in use*. This is achieved by hardware-based memory encryption; in addition to isolation among VMs, data is decrypted only when it is in use by CPU. This prevents other cloud tenants and even the cloud provider from accessing the data used by the confidential VMs, which can further improve the

privacy of vehicle data utilized by the oracle model for ground truth generation or during the retraining process.

- Update overhead: The lightweight nature of the device model allows us to run both the retrained and existing models simultaneously for a smooth transition between them. We executed two instances of the SSD-MobileNet model simultaneously on the NVIDIA Jetson Orin, which led to a less than 1 ms increase in the inference times for each of the model instances. Specifically, a single model instance takes an average of 4.7 ms (with a standard deviation of 0.6 ms) per frame while the concurrent execution results in the model taking 5.03 ms (with a standard deviation of 0.37 ms) per frame. Pruning, quantization, and hardware-specific optimizations help achieve these extremely low inference times.
- Model agnostic framework: The proposed framework is not limited to Faster R-CNN or any particular models as an oracle. Using more complex architectures like vision transformers [28] [29] can provide more accurate ground truth and thus improve the retraining performance.
- Other ML applications: A highly accurate oracle model is crucial for leveraging the proposed framework, as it fully automates the retraining pipeline to continually improve the vehicle model. Such powerful models may not be easily available for public use in certain ML applications such as lane detection, steering angle prediction, and segmentation. Furthermore, most AV datasets are heavily focused on the object detection task. Therefore, we limit our evaluation to object detection tasks, which are critical in AVs and for which highly accurate models are more readily available.

V. Conclusion

AdaptAV continuously improves the vision model on an autonomous vehicle by leveraging the cloud. The cloud runs a highly accurate, large oracle model that is used to guide the retraining of a small, on-vehicle perception model using the live camera stream uploaded by the vehicle. We also presented frame sampling techniques that can reduce the chance of model overfitting and also improve the prediction accuracy. An intuitive extension of this work is to expand the scale of the system by collecting image streams from multiple vehicles. This will lead to a more robust model that can generalize better to diverse novel scenarios. We leave this as future work.

Acknowledgment

This work is supported in part by NSF grant 2302610, NCSU Faculty Research and Professional Development, and the Google Cloud Research Credits program with the award GCP19980904. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of sponsors.

References

[1] H. Lipson and M. Kurman, *Driverless: intelligent cars and the road ahead*. Mit Press, 2017.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, 2015.

[3] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Proc. of the European Conference on Computer Vision*, 2016.

[5] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[6] "Self-driving uber kills arizona woman in first fatal crash involving pedestrian," *The Guardian*, 2018.

[7] "Waymo and zoox are under federal investigation as self-driving cars behave erratically," *CNN*, May 2024.

[8] Tesla, "Update vehicle firmware to improve certain fsd beta driving operations," 2024.

[9] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[10] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[11] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[12] R. Girshick, "Fast r-cnn," in *Proc. of the IEEE International conference on Computer Vision*, 2015.

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proc. of the IEEE international conference on computer vision*, 2017.

[14] H.-J. Jeong, H.-J. Lee, C. H. Shin, and S.-M. Moon, "Ionn: Incremental offloading of neural network computations from mobile devices to edge servers," in *Proc. of the ACM symposium on cloud computing*, 2018.

[15] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "Spinn: synergistic progressive inference of neural networks over device and cloud," in *Proc. of the 26th annual international conference on mobile computing and networking*, 2020.

[16] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge ai: On-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, 2019.

[17] S. Lu, X. Yuan, and W. Shi, "Edge compression: An integrated framework for compressive imaging processing on cavs," in *2020 IEEE/ACM Symposium on Edge Computing*, 2020.

[18] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *19th USENIX Symposium on Networked Systems Design and Implementation*, 2022.

[19] M. Khani, P. Hamadanian, A. Nasr-Esfahany, and M. Alizadeh, "Real-time video inference on edge devices via adaptive model streaming," in *Proc. of the IEEE/CVF International Conference on Computer Vision*, 2021.

[20] "About ffmpeg," https://ffmpeg.org/about.html.

[21] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[22] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. of the IEEE International Conference on Image Processing*, 2016.

[23] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-camera filtering for resource-efficient real-time video analytics," in *Proc. of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020.

[24] "Nvidia jetson orin," https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/.

[25] "Cloud computing services — google cloud," https://cloud.google.com.

[26] "Zed 2," https://www.stereolabs.com/products/zed-2.

[27] "Google cloud confidential vm overview," https://cloud.google.com/confidential-computing/confidential-vm/docs/confidential-vm-overview.

[28] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[29] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. of the European conference on computer vision*, 2020.